CVSD Final Project

Team13 Report

R12921A01 周固廷 R12943112 謝承恩

- Algorithm
  - QR decomposition algorithm introduction

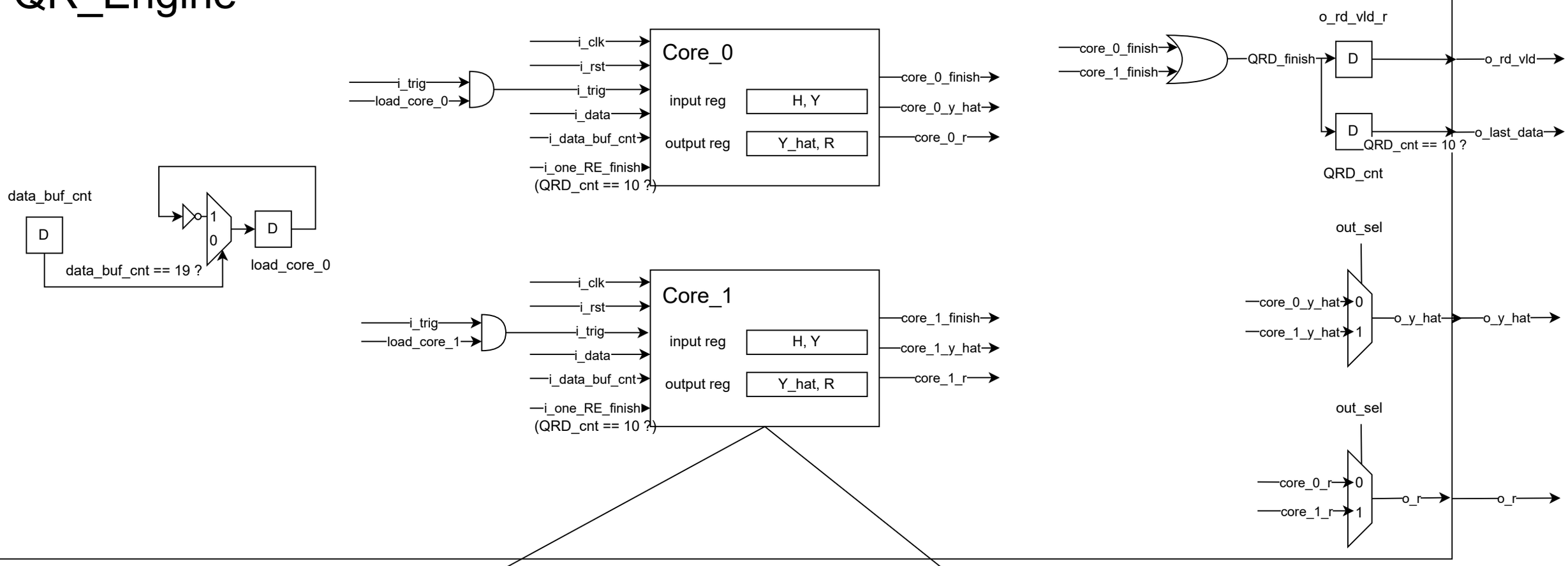
    我們使用的是投影片提供的演算法，也就是 Modified Gram-Schmidt，這個方法每個回合會計算 column vector 的長度，和 column vector 與正規化後的正交向量之內積，再將正交向量減去自己在 h1 的正射影長，總共需做 4 個回合，最後在加正規化後的正交向量和輸入 Y 向量相乘得 y_hat。

  - FXP setting
    1. 首先將 input data truncate 成 S1.11 的格式(丟棄後 11 個 bit)
    2. 在計算內積、平方、純量乘法等運算時，所有的乘法器都是 13 bits * 13 bits 的乘法器，因此乘積是 26bits (S3.22)，然後再將乘積 truncate 成 S3.16 的格式(丟棄後 6 個 bits)，再進行相加，輸出的格式為 S3.16，不特別考慮 overflow 的情況，因為對整體的 error rate 影響不大。

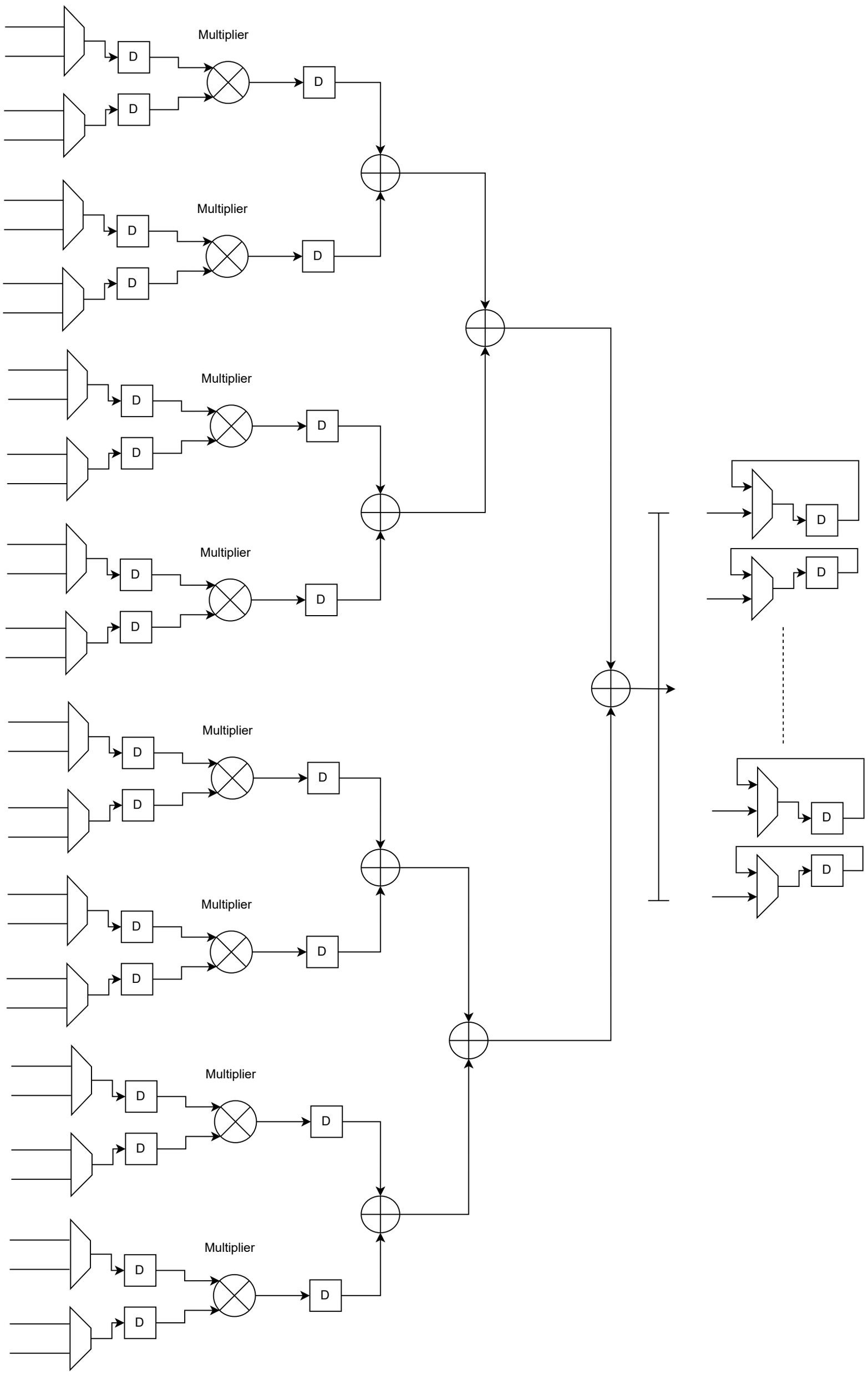
- HW implementation
  - HW block diagram (見下一頁)

    由於要將 clock period 壓到 5ns，因此運過程中需要切很多 pipeline，經過評估後我們無法在 20 個 cycle 算完一筆資料，因此我們使用 parallel 的技巧，用兩套相同的硬體，每套硬體 40 個 cycle 會輸出 1 筆資料。

QR_Engine

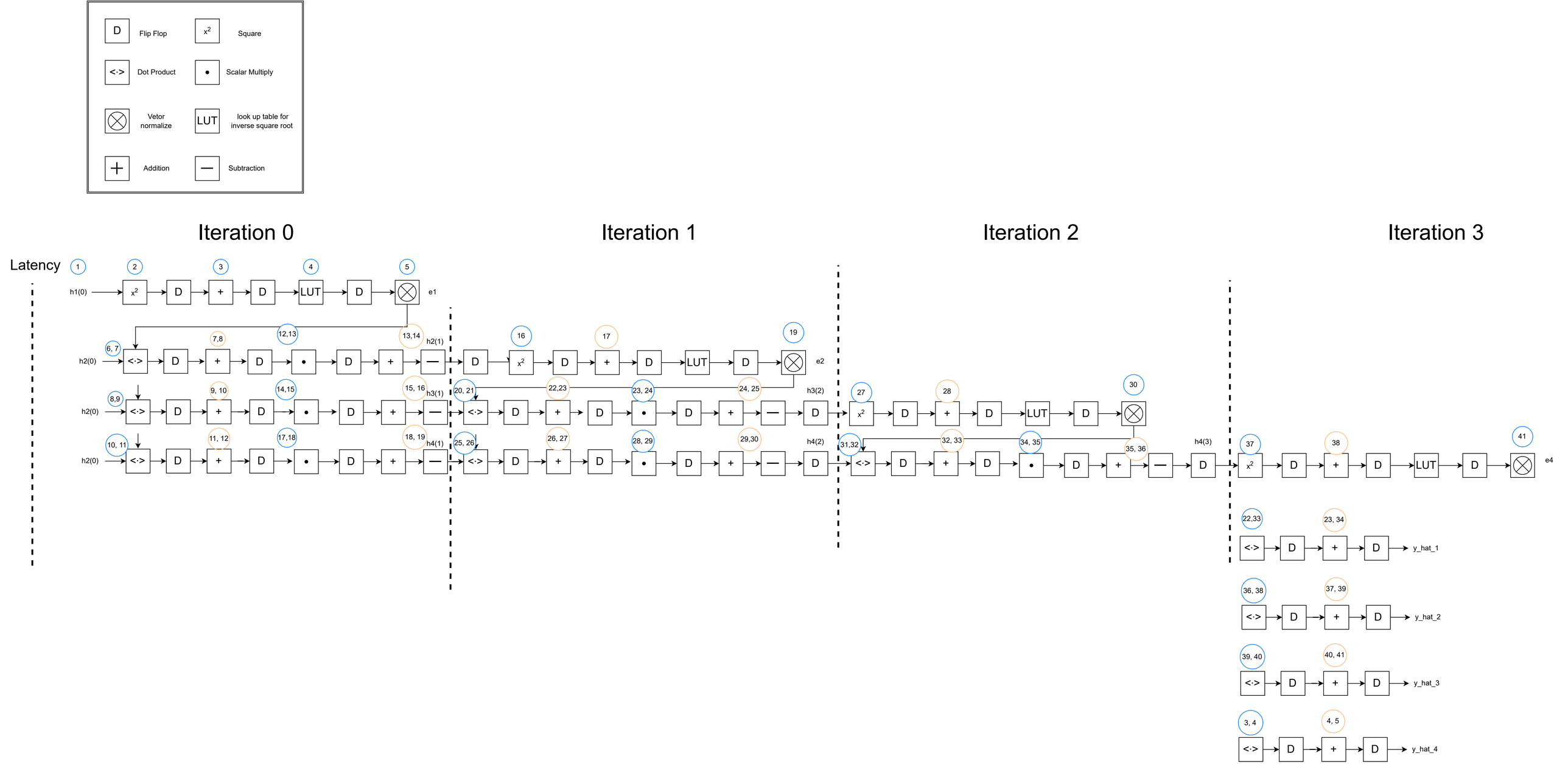


Core



- HW scheduling (見下一頁)

  運算資源最主要是乘法器和加法器，而 Modified Gram-Schmidt 一次的運算總共需要 324 個乘法，而我們的一個 core moudle 每 40 個 cycle 輸出一筆資料，因此平均到每個 cylcle 後是 8.多個乘法，也就是至少需要 9 個乘法器，因此一個 core 我們使用了 9 個乘法器，加法器則使用 7 個用來將八個乘積做相加，藍色圈圈內的數字是乘法器的排程，橘色圈圈為加法器的排程，圈圈內之數字為 latency，範圍是 2～41，
  (latency 1 為預備狀態)。


D
Flip Flop
x2
Square
<·>
Dot Product
•
Scalar Multiply
Vetor normalize
LUT
look up table for inverse square root
+
Addition
—
Subtraction
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Latency
h1(0)
h2(0)
h2(0)
h2(0)
e1
e2
e4
h2(1)
h3(1)
h4(1)
h3(2)
h4(2)
h4(3)
y_hat_1
y_hat_2
y_hat_3
y_hat_4

- Area / Power / Latency report
  - Area

```
innovus 19> analyzeFloorplan
**WARN: (IMPAFPU-9006): Command 'analyzeFloorplan' is obsolete. Please use commands 'place
_timing_debug_report -proto' to analyze congestion and timing for the floorplan.
Start to collect the design information.
Build netlist information for Cell QR_Engine.
Finished collecting the design information.
Average module density = 1.000.
Density for the design = 1.000.
       = stdcell_area 380618 sites (646061 um^2) / alloc_area 380618 sites (646061 um^2).
Pin Density = 0.3257.
            = total # of pins 123964 / total area 380618.
************************ Analyze Floorplan *************************
    Die Area(um^2)             : 712519.86
    Core Area(um^2)            : 646060.99
    Chip Density (Counting Std Cells and MACROs and IOs): 90.673%
    Core Density (Counting Std Cells and MACROs): 100.000%
    Average utilization        : 100.000%
    Number of instance(s)      : 62631
    Number of Macro(s)         : 0
    Number of IO Pin(s)        : 533
    Number of Power Domain(s) : 0
************************ Estimation Results ************************
********************************************************************
```

  - Power

```
  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group

                          Internal  Switching  Leakage    Total
Power Group               Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------------------
clock_network           6.174e-03 1.192e-03 2.987e-05 7.396e-03 (20.89%)
register                8.124e-03 5.548e-04 1.272e-04 8.806e-03 (24.87%)  i
combinational              0.0120 6.954e-03 2.398e-04    0.0192 (54.24%)
sequential                 0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                     0.0000    0.0000    0.0000    0.0000 ( 0.00%)
io_pad                     0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                  0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 8.701e-03   (24.58%)
  Cell Internal Power  =    0.0263   (74.30%)
  Cell Leakage Power   = 3.969e-04   ( 1.12%)
                         ---------
Total Power            =    0.0354  (100.00%)

X Transition Power     = 3.252e-05
Glitching Power        = 7.128e-05

Peak Power             =    0.6008
Peak Time              =     5.607
```

  - Latency

```
Pattern  901# ~ 1000# are written
$finish called from file "./testfixture-3.v", line 260.
$finish at simulation time           121516500
         V C S   S i m u l a t i o n   R e p o r t
Time: 121516500 ps
CPU Time:    196.910 seconds;       Data structure size:   8.8Mb
Tue Dec 19 19:05:46 2023
CPU time: 15.216 seconds to compile + 4.537 seconds to elab + 2.631 seconds to link + 196.963 seconds in simulation
*** Using c compiler gcc instead of cc ...
                    Chronologic VCS (TM)
        Version T-2022.06_Full64 -- Tue Dec 19 19:05:53 2023
```

◆ Technique sharing for HW improvement

避免使用開根號電路和除法器：

假設a為一複數向量，$< a, a >$是a向量自己和自己做內積運算。

由於 Modified Gram-Schmidt 最大的瓶頸在於求向量長度時需要開根號，正規化向量時需要做除法($a \div \sqrt{< a, a >}$)，因此我們首先想到可以使用近似解(例如牛頓勘根法等方式)求出實部與虛部平方的加總之開根號倒數，也就是 $\frac{1}{\sqrt{<a,a>}}$，若求出了這個近似解，在做向量正規化時，就可以只用乘法完成($a \times \frac{1}{\sqrt{<a,a>}}$)。

求出了$\frac{1}{\sqrt{a}}$的近似解以後，我們也可以用它來求出向量長: $\sqrt{< a, a >}$，只要將$< a, a >$ 乘 s上剛剛求出的近似解($< a, a > \times \frac{1}{\sqrt{<a,a>}}$) 即可得到$\sqrt{< a, a >}$。

但是使用牛頓勘根法仍有問題需要考慮，首先是如何找到一個好的初始值，若初始值沒有找好，則精確度並不高，此外，逼近的過程需要迭代多少次也是需要考慮的一個問題，因此初始值的精度要設為多少和迭代次數之間的取捨，理論上需要透過軟體模擬的方式，才可以得到比較好的結論，然而即便解決了上述問題後，牛頓勘根法仍需要做 3 次的乘法，此外，乘積的 bits 數要取多少也是一個很大的問題，當分母很小的時候，$\frac{1}{\sqrt{<a,a>}}$的值就會變得很大，基於以上理由，我們最終放棄使用近似法，同時我們也嘗試使用過 design ware 的 inv_sqrt 模組，然而我們實際使用時發現這個 ip 的 latency 和 area 都非常大，猜測也許內部仍然是使用開根號和除法來實作，因此也放棄使用這個 ip。我們最終使用的是查表法，由於我們的 $< a, a >$ 有 18 個 bits (S6.11)，若想直接針對這個長度製作 look up table 基本上是不可行的，但可以利用數學性質的輔助來減少 look up table，當 $< a, a >$ 落在 [$\frac{1}{4}$, 1] 的區間時，$\frac{1}{\sqrt{<a,a>}}$ 的值會落在(1, 2) 之間，因此我們的作法是先將 $< a, a >$ normalize 到 [$\frac{1}{4}$, 1] 這個區間(利用 shift，將 leading one shift 到 a_1 或 a_2 的位置，shift 到哪裡是我們決定好的)，也就是 $< a, a >'$ = (0.$a_1a_2a_3a_4a_5$...a_{10})，其中 (a_1, a_2) 這兩個 bit 必須至少其中一個 bit 是 1，接下來就對 $< a, a >'$ 進行查表，因為小數點後只剩下 10 個 bits，因此 look up table 只要 1024 個 entry，而 look up table 的生成就是直接使用軟體計算根號倒數的值。因此在硬體上的實現我們需要一個 leading zero 的電路，以及 shifter 將 a shift 成 $< a, a >'$ 的形式。

需要注意的是，查表前 shift 的次數必須為偶數，因為根號是將冪次除以 2 (也就是$\frac{1}{2}$次方)。在查表之後，也要記得將數字 inverse normalize 回去(將數字反過來 shift 回去)，因為中途有經過根號運算，所以要 shift 的次數是做查表前的$\frac{1}{2}$，因此原本的冪次必須是偶數，否則會丟失 LSB。

透過這個方法我們在沒有使用乘法器、除法器的情況下求出根號倒數的值。